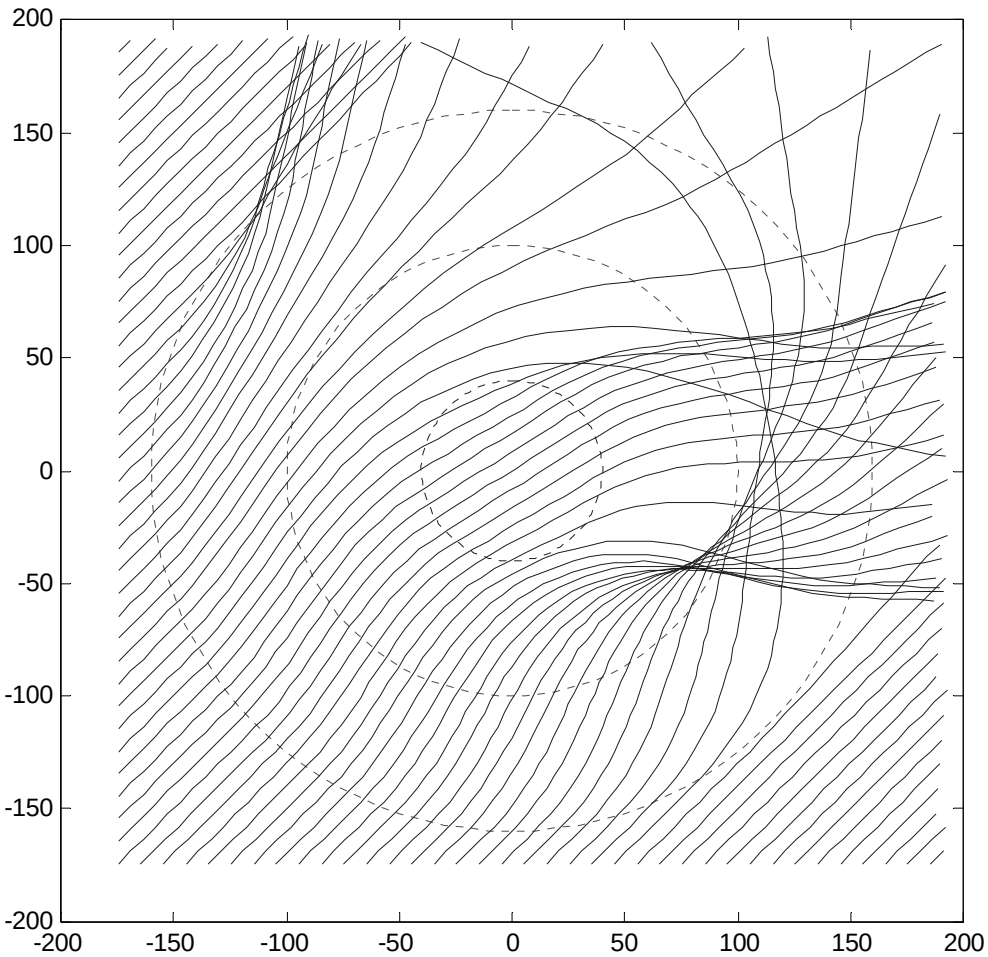
Code for Rogue Wave circular current test problem

```
% This .m-file circular_current computes ray paths through a current field. As
% an example, we use the circular current example in the modeling book.
close all;
global X Y U V UX UY VX VY

% Define the computational domain
n = 80;                             % number of grid intervals in x- and y-
                                    % directions used to represent the current
de = 200e3;                         % field over a physical domain extending
[X,Y] = meshgrid(linspace(-de, de,n+1));  % X and Y coordinates at node points

% Set up the circular current field U, V
ri = 40e3; ro = 160e3;              % Edges of current at dist. ri and ro resp.
rc = (ri+ro)/2;
sp = 2;                             % Current speed 2 m/s.
U = zeros(n+1); V = zeros(n+1);     % Reserve space for the velocity fields
av = zeros(n+1);
r       = sqrt(X.^2+Y.^2);          % Find distance to center and note
cur     = r>ri & r<ro;              % where the current is present
av(cur) = sp*4*(r(cur)-ri).*(r(cur)-ro)./(r(cur)*(ro-ri)^2);
U(cur)  = -av(cur).*Y(cur);         % Set the velocity field of the current
V(cur)  =  av(cur).*X(cur);
h = 2*de/n;                         % Grid spacing
[UX,UY] = gradient(U,h); [VX,VY] = gradient(V,h);   % Calculate UX UY VX VY

% Define physical parameters
g = 9.8;                            % Acceleration of gravity
t = 10;                             % Time period of incoming waves (in seconds)
s = 2*pi/t;                         % Angular time frequency
k = s^2/g;                          % Magnitude of initial wave vector

% Plot the circular current
ang = linspace(0,2*pi,101); si = sin(ang)*1e-3; co = cos(ang)*1e-3;
plot(ri*si,ri*co,':',rc*si,rc*co,':',ro*si,ro*co,':')
axis ([-200 200 -200 200]); axis square; hold on;

% Loop over the rays
dt = 1000;                          % Specify time step (in seconds).
for lt = -360:10:360
   lt                               % Show progress of loop on the screen
   xykk(1) = max( lt-180,-180)*1e3; % Give start x- and y-locations for the
   xykk(2) = max(-lt-180,-180)*1e3; % rays shown in Figure I2.4-3.
   xykk(3) = k/sqrt(2);             % Component in x-dir (45 degree)
   xykk(4) = k/sqrt(2);             % Component in y-dir (45 degree)

   % Run ODE for ray until it exits the domain, using 2-stage second order RK
   ct = 1; clear x y;
   while 1                                  % Keep stepping until stop criterion
      rhs_xykk = c_rhs(xykk);               % RK stage 1 ; evaluate RHS
      if any(isnan(rhs_xykk)),break,end     % If reach edge, quit ray
      rhs_xykk = c_rhs(xykk+dt/2*rhs_xykk); % RK stage 2 ; evaluate RHS
      if any(isnan(rhs_xykk)),break,end     % If reach edge, quit ray
      xykk  = xykk + dt*rhs_xykk;           % Update vector with x,y,k1,k2
      x(ct) = xykk(1); y(ct) = xykk(2);     % Store x,y-values to plot the ray.
      ct = ct+1;
   end

   % Plot the ray
   plot(x(:)*1e-3,y(:)*1e-3)           % Scale down by 1e-3 to get unit km.
end
```

```
function rhs_xykk = c_rhs(xykk)
% Given x,y,k1,k2 in the vector xykk; calculate the RHS of the ODE system
global X Y U V UX UY VX VY
x  = xykk(1); y  = xykk(2); k1 = xykk(3); k2 = xykk(4);
if max(abs(x),abs(y))>X(1,end-2)       % If edge is approached, signal that
    rhs_xykk = [NaN NaN NaN NaN];      % information back through NaN's
else
    u  = interp2(X,Y,U ,x,y,'cubic');
    v  = interp2(X,Y,V ,x,y,'cubic');
    ux = interp2(X,Y,UX,x,y,'cubic');
    uy = interp2(X,Y,UY,x,y,'cubic');
    vx = interp2(X,Y,VX,x,y,'cubic');
    vy = interp2(X,Y,VY,x,y,'cubic');
    g = 9.8 ;                          % With u,v,ux,uy,vx,vy known at the
    k = sqrt(k1^2+k2^2);               % location x,y, we can calculate the
    alpha = 0.5*sqrt(g/k^3);           % four components of the RHS
    rhs_xykk = [alpha*k1+u, alpha*k2+v, -(k1*ux+k2*vx), -(k1*uy+k2*vy)];
end
```